ELSEVIER

Journal of Computational and Applied Mathematics 123 (2000) 217–240

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Iterative methods for large continuation problems

D. Calvetti[a, *, 1], L. Reichel[b, 2]

[a] *Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106, USA*

[b] *Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242, USA*

Received 1 September 1999; received in revised form 11 September 1999

Abstract

The computation of solution paths for continuation problems requires the solution of a sequence of nonlinear systems of equations. Each nonlinear system can be solved by computing the solution of a succession of linear systems of equations determined by Jacobian matrices associated with the nonlinear system of equations. Points on the solution path where the Jacobian matrix is singular are referred to as singular points and require special handling. They may be turning points or bifurcation points. In order to detect singular points, it is essential to monitor the eigenvalues of smallest magnitude of the Jacobian matrices generated as the solution path is traversed. We describe iterative methods for the computation of solution paths for continuation problems so large that factorization of the Jacobian matrices is infeasible or impractical. The iterative methods simultaneously solve linear systems of equations determined by the Jacobian matrices and compute a few eigenvalue–eigenvector pairs associated with the eigenvalues of smallest magnitude of each Jacobian. A bordering algorithm with a pseudo-arclength parametrization is applied in the vicinity of turning points to overcome the singularity of the Jacobian. A bifurcation perturbation strategy is used to compute solution paths at bifurcation points. Our iterative methods are based on the block-Lanczos algorithm and are applicable to problems with large symmetric Jacobian matrices.

MSC: 65F15

Keywords: Path following; Turning point; Bifurcation point; Eigenvalue computation; Bordering algorithm; Nonlinear system

1. Introduction

Many problems in science and engineering require the computation of a family of solutions $u(\lambda) \in \mathbb{R}^n$, $a \leqslant \lambda \leqslant b$, of a nonlinear system of equations of the form

* Corresponding author.

E-mail addresses: dxc57@po.cwru.edu (D. Calvetti), reichel@mcs.kent.edu (L. Reichel).

[1] Research supported in part by NSF grant DMS-9806702.

[2] Research supported in part by NSF grants DMS-9806413 and ASC-9720221.

0377-0427/00/$ - see front matter
PII: S0377-0427(00)00405-2

$$G(u,\lambda)=0, \quad u=u(\lambda), \tag{1.1}$$

where $G:\mathbb{R}^n\times\mathbb{R}\to\mathbb{R}^n$ is a continuously differentiable function of $u\in\mathbb{R}^n$ and $\lambda\in\mathbb{R}$. The parameter λ is often a quantity of physical significance, such as temperature in liquid crystal modeling [4] or the Reynolds number in hydrodynamical flow [15], and is commonly referred to as the "natural parameter". We are interested in determining solution paths

$$\Gamma:=\{(u,\lambda):\ G(u,\lambda)=0, u=u(\lambda), a\leqslant\lambda\leqslant b\}$$

associated with (1.1). Here a and b are given bounds for λ.

The solutions $u(\lambda)$, $a\leqslant\lambda\leqslant b$, of (1.1) are commonly computed by a continuation method. In these methods an initial value problem for u is derived by differentiating Eq. (1.1) with respect to λ. Thus, let $u=u(\lambda)$ satisfy (1.1). Then differentiation of (1.1) yields

$$G_u(u(\lambda),\lambda)\dot{u}(\lambda)+G_\lambda(u(\lambda),\lambda)=0, \tag{1.2}$$

where $\dot{u}=\mathrm{d}u/\mathrm{d}\lambda$. Given $u(a)$ and assuming that the Jacobian matrix G_u is nonsingular in a neighborhood of the solution path, we can compute $u(\lambda)$ for $a<\lambda\leqslant b$ by solving the initial value problem (1.2) for $u=u(\lambda)$. Points where the Jacobian matrix $G_u(u,\lambda)$ is nonsingular are referred to as regular points; points where $G_u(u,\lambda)$ is singular are referred to as singular points. Singular points on the solution path are either turning points or bifurcation points of the solution path. The determination of the solution path in a neighborhood of a turning point or bifurcation point requires special care. It is therefore important to detect singular points on the solution path.

This paper describes new algorithms for path following. The algorithms are designed to be applicable to problems so large that factorization of the Jacobian matrices into triangular or orthogonal factors is unfeasible or undesirable. Our algorithms only evaluate matrix–vector products with the Jacobian matrices. Therefore, only a few of the nonvanishing entries of each Jacobian matrix generated have to be stored in fast computer memory simultaneously; the entries can be computed as they are required for the evaluation of matrix–vector products and discarded when they are not needed. This approach requires little computer memory and is therefore well suited for large problems, such as the liquid crystal modeling problem discussed in [4].

We assume that the Jacobian matrices are symmetric. Our algorithms are based on an iterative method for the simultaneous solution of linear systems of equations with the Jacobian matrix and computation of a few of the eigenvalues of smallest magnitude and associated eigenvectors of the Jacobian. Since the Jacobian is singular at turning and bifurcation points on the solution path and regular elsewhere on the solution path, knowledge of the eigenvalue closest to the origin makes it easy to identify these points. Moreover, the eigenvectors associated with the smallest eigenvalues are helpful for path following in the vicinity of a turning or bifurcation point.

Our iterative method for solving linear systems, while simultaneously computing a few eigenvalue–eigenvector pairs, is based on the implicitly restricted block-Lanczos (IRBL) method introduced in [4]. This method is a block-variant of the implicitly restarted Lanczos method discussed in [2,6,17].

Bifurcation points are traversed by two or more solution paths. Different methods for continuing paths across bifurcation points have been proposed in the literature. This paper only discusses the "perturbed bifurcation" method, where a small perturbation of Eq. (1.1) is introduced at a bifurcation point. This makes a bifurcation point split into close regular or turning points; see Georg [13] for a discussion and illustrations.

Continuation methods for path following have received considerable attention. A nice survey of the mathematical background is provided by Keller [16]. Only few algorithms are available for large-scale problems; see, e.g., [1,8,9,14,15,18]. Our algorithms differ from those available in the literature in that they are based on the IRBL method and are designed to be applicable for problems with very large symmetric Jacobian matrices.

This paper is organized as follows. Section 2 recalls a few useful results on the solution of continuation problems and discusses the calculations needed for path following. In Section 3, we outline an iterative method, previously introduced in [5], for the computation of a few eigenpairs of a large symmetric matrix, and the simultaneous solution of a linear system of equations with this matrix. Section 4 describes how to apply this iterative method to path following in the presence of turning and bifurcation points. We present path following algorithms based on the Euler–Newton predictor-corrector scheme for use at regular points on the solution path. A pseudo-arclength parametrization is used in the vicinity of turning points. Numerical examples are presented in Section 5 and concluding remarks can be found in Section 6.

2. An overview of the path following problem

The first part of this section reviews the continuation problem and introduces notation to be used in the remainder of the paper. In the second part the computational problems are described.

2.1. Theory

In this subsection the Jacobian matrix is allowed to be nonsymmetric. We focus on the interplay between geometry and computations. An excellent introduction to the numerical analysis of continuation problems is provided by Keller [16] and much of our discussion follows his presentation.

The purpose of a continuation method is to determine solutions of problem (1.1) for all λ in a specified interval $[a,b]$. Let $\lambda^1=\lambda^0+\Delta\lambda$ with $\lambda^0,\lambda^1\in[a,b]$ and assume that the solution $u^0=u(\lambda^0)$ of (1.1) for $\lambda=\lambda^0$ is known. The Implicit Function Theorem provides the theoretical basis for computational methods for determining the solution $u^1=u(\lambda^1)$ of (1.1) for $\lambda=\lambda^1$ when $\Delta\lambda$ is of sufficiently small magnitude.

Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm or the induced matrix norm. We note, however, that when G stems from the discretization of a differential or integral equation, it can be advantageous to select a norm that depends on the discretization; see Ferng and Kelley [12] for a discussion.

Introduce the sets

$$B_\rho(u^0):=\{u\in\mathbb{R}^n:\ \|u-u^0\|<\rho\},\qquad B_\rho(\lambda^0):=\{\lambda\in\mathbb{R}:\ |\lambda-\lambda^0|<\rho\}$$

for $\rho>0$.

Theorem 2.1 (Implicit Function Theorem). *Let $G:\mathbb{R}^n\times\mathbb{R}\to\mathbb{R}^n$ be a function, such that for some sufficiently small constants $\rho_1>0$ and $\rho_2>0$,*

(i) *$G(u^0,\lambda^0)=0$ for some $u^0\in\mathbb{R}^n$ and $\lambda^0\in\mathbb{R}$,*

(ii) *G and G_u are continuous in $B_{\rho_1}(u^0) \times B_{\rho_2}(\lambda^0)$,*
(iii) *$G_u(u^0, \lambda^0)$ is nonsingular with a bounded inverse.*

Then for every $\lambda \in B_{\rho_2}(\lambda^0)$ there is a unique $u := u(\lambda) \in B_{\rho_1}(u^0)$, such that

(a) *$G(u, \lambda) = 0$ and $u(\lambda^0) = u^0$,*
(b) *$u = u(\lambda)$ is a continuous function of λ on $B_{\rho_2}(\lambda^0)$.*

Proof. The theorem can be formulated for u and λ in more general sets than $\mathbb{R}^n$ and $\mathbb{R}$, respectively. For instance, Keller [16, Section 2.7] presents a proof when u belongs to a Banach space and λ to a parameter space. □

It is an immediate consequence of the Implicit Function Theorem that the continuation problem for (1.1) has a unique solution in a neighborhood of a regular point (u^0, λ^0) on the solution path.

Given a regular point (u^0, λ^0) for (1.1), the solution $u^1 = u(\lambda^1)$ of (1.1) for $\lambda = \lambda^1$ can be computed by a predictor–corrector scheme when $\Delta\lambda = \lambda^1 - \lambda^0$ is of sufficiently small magnitude. The predictor determines an initial approximation $u_0(\lambda^1)$ of the solution $u(\lambda^1)$ of (1.1). It follows from Theorem 2.1 that for some $\rho_2 > 0$ and every $\lambda \in B_{\rho_2}(\lambda^0)$, there is a unique $u(\lambda) \in \mathbb{R}^n$, such that

$$G(u(\lambda), \lambda) = 0. \tag{2.1}$$

Differentiating (2.1) with respect to λ yields (1.2). Substituting $\lambda = \lambda^0$ into G_u and G_λ in (1.2) gives the linear system of equations

$$G_u(u^0, \lambda^0)\dot{u}^0 = -G_\lambda(u^0, \lambda^0) \tag{2.2}$$

for $\dot{u}^0 = \dot{u}(\lambda^0)$. Application of Euler's method as a predictor yields the approximation

$$u_0(\lambda) := u^0 + (\lambda - \lambda^0)\dot{u}^0 \tag{2.3}$$

of $u(\lambda)$. The error in this approximation is given by

$$u(\lambda) - u_0(\lambda) = \tfrac{1}{2}\ddot{u}(\lambda^0)(\lambda - \lambda^0)^2 + \mathcal{O}((\lambda - \lambda^0)^3),$$

which reflects that Euler's method is an integration method of order one. Here $\ddot{u} = \mathrm{d}^2u/\mathrm{d}\lambda^2$.

In general, $(u_0(\lambda), \lambda)$ does not satisfy (1.1). It is convenient to use Newton's method for (1.1) as a corrector. The iterates u_k, $k = 1, 2, \ldots$, determined by Newton's method are given by

$$u_{k+1} := u_k + \Delta u, \quad k = 0, 1, \ldots, \tag{2.4}$$

where Δu solves the linear system of equations

$$G_u(u_k, \lambda)\Delta u = -G(u_k, \lambda). \tag{2.5}$$

Assume that the conditions of Theorem 2.1 are satisfied at (u^0, λ^0). Then G_u is continuous and invertible in a neighborhood of (u^0, λ^0). It can be shown that for a step size $\Delta\lambda$ of sufficiently small magnitude, the iterates determined by the Euler–Newton predictor–corrector scheme converge to $u(\lambda)$. The following definitions are helpful for the analysis of the continuation problem for (1.1) at points where the Jacobian matrix is singular.

Definition 2.2. Let $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ be a continuously differentiable function. A point $(u(\lambda), \lambda)$ on the solution path is said to be a simple singular point if $G_u = G_u(u(\lambda), \lambda))$ is singular and

$$\dim \mathcal{N}(G_u) = 1, \tag{2.6}$$

where $\mathcal{N}(G_u)$ denotes the null space of G_u.

Definition 2.3. A simple singular point $(u(\lambda), \lambda)$ on a solution path Γ is said to be a turning point if

$$G_\lambda \notin \mathcal{R}(G_u) \tag{2.7}$$

and a bifurcation point if

$$G_\lambda \in \mathcal{R}(G_u), \tag{2.8}$$

where $\mathcal{R}(G_u)$ denotes the range of $G_u = G_u(u(\lambda), \lambda)$, and $G_\lambda = G_\lambda(u(\lambda), \lambda)$.

In this paper we only consider turning points and bifurcation points that are simple singular points. We refer to Decker and Keller [10], Georg [13] and Keller [16] for discussions on more general singular points.

We first consider turning points. It is convenient to introduce the arclength parameter s of Γ. Henceforth, we write $u = u(s)$, $\lambda = \lambda(s)$ and $G(s) = G(u(s), \lambda(s))$, and the derivatives $\dot{u}$ and $\dot{\lambda}$ denote differentiation with respect to s. We have

$$\|\dot{u}\|^2 + \dot{\lambda}^2 = 1, \tag{2.9}$$

which shows that the tangent vector $(\dot{u}(s), \dot{\lambda}(s))$ of Γ is of unit length.

Assume that $(u(s^0), \lambda(s^0))$ is a turning point. Differentiating $G(s) = 0$ with respect to s yields, analogously to (1.2),

$$G_u(s)\dot{u}(s) + G_\lambda(s)\dot{\lambda}(s) = 0. \tag{2.10}$$

Proposition 2.4. *Let $(u(s^0), \lambda(s^0))$ be a simple turning point on the solution path Γ. Then*

$$\dot{\lambda}(s^0) = 0, \quad \dot{u}(s^0) \in \mathcal{N}(G_u(s^0)). \tag{2.11}$$

Proof. Assume that $\dot{\lambda}(s^0) \neq 0$. Then

$$G_\lambda(s^0) = -\frac{G_u(s^0)\dot{u}(s^0)}{\dot{\lambda}(s^0)},$$

which contradicts (2.7). Substituting $\dot{\lambda}(s^0) = 0$ into (2.10) yields $G_u(s^0)\dot{u}(s^0) = 0$ and the proposition follows. □

The null spaces of $G_u(s^0)$ and $G_u^{\mathrm{T}}(s^0)$ are of the same dimension. For future reference we introduce basis vectors ϕ and ψ of these spaces, i.e.,

$$\mathcal{N}(G_u(s^0)) = \operatorname{span}\{\phi\}, \quad \mathcal{N}(G_u^{\mathrm{T}}(s^0)) = \operatorname{span}\{\psi\}, \quad \|\phi\| = \|\psi\| = 1. \tag{2.12}$$

Since $G_u(s)$ is singular at a turning point, Newton's method cannot be applied with arclength parameterization to continue the solution path across a turning point. This difficulty can be overcome

by imposing an additional constraint. Recall that the unit tangent vector of Γ at $(u(s^0), \lambda(s^0))$ is given by $(\dot{u}(s^0), \dot{\lambda}(s^0))$, cf. (2.9). The equation of a plane orthogonal to the unit tangent at a distance Δs from the point (u^0, λ^0) is given by $N(u, \lambda, \Delta s) = 0$, with

$$N(u, \lambda, \Delta s) := \dot{u}^{0\mathrm{T}}(u - u^0) + \dot{\lambda}^0(\lambda - \lambda^0) - \Delta s.$$

This plane intersects the path Γ provided that the curvature of Γ at (u^0, λ^0) or Δs are sufficiently small. Thus, the point of intersection between the path and the plane satisfies the nonlinear system of equations

$$G(u, \lambda) = 0, \tag{2.13}$$

$$N(u, \lambda, \Delta s) = 0. \tag{2.14}$$

The solution of these equations by Newton's method yields iterates

$$(u_{k+1}, \lambda_{k+1}) := (u_k + \Delta u, \lambda_k + \Delta\lambda),$$

where Δu and $\Delta\lambda$ satisfy the linear system of equations

$$\begin{bmatrix} G_u^k & G_\lambda^k \\ \dot{u}^{0\mathrm{T}} & \dot{\lambda}^0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -G^k \\ -N^k \end{bmatrix}. \tag{2.15}$$

Here and below, we use the notation

$$\begin{aligned} &G^k = G(u_k, \lambda_k), \quad G_u^k = G_u(u_k, \lambda_k), \quad G_\lambda^k = G_\lambda(u_k, \lambda_k), \\ &N^k = N(u_k, \lambda_k, \Delta s), \quad \dot{u}^0 = \dot{u}(s^0), \quad \dot{\lambda}^0 = \dot{\lambda}(s^0). \end{aligned} \tag{2.16}$$

The solution of (2.15) exists and is unique even if G_u^k is singular, provided that the matrix

$$\tilde{G}^k := \begin{bmatrix} G_u^k & G_\lambda^k \\ \dot{u}^{0\mathrm{T}} & \dot{\lambda}^0 \end{bmatrix} \tag{2.17}$$

is nonsingular. Necessary and sufficient conditions for $\tilde{G}^k$ to be nonsingular are discussed in Section 2.2 below.

Following Keller [16], we refer to the parameterization which uses the plane normal to the path as pseudo-arclength parameterization. Although the pseudo-arclength parameterization is usually applied to singular points, it can be used also at regular points.

Constraint (2.14) is advocated by Keller [16] and will be used in the numerical examples of Section 5. Other constraints, among them Eq. (2.38) below, have also been proposed in the literature; see e.g., [1,15] for discussions.

We turn to bifurcation points (u^0, λ^0) of the solution path and assume that the Jacobian $G_u = G_u(u^0, \lambda^0)$ satisfies (2.6) and (2.8).

Proposition 2.5. *Let (u^0, λ^0) be a bifurcation point on the solution path. Then*

$$\mathcal{N}([G_u^0, G_\lambda^0]) = \operatorname{span}\left\{ \begin{bmatrix} \phi \\ 0 \end{bmatrix}, \begin{bmatrix} v \\ 1 \end{bmatrix} \right\},$$

where ϕ satisfies (2.12) *and v is the unique vector determined by*

$$G_u^0 v = -G_\lambda^0, \quad \phi^{\mathrm{T}} v = 0. \tag{2.18}$$

Moreover, for some $\alpha \in \mathbb{R}$,

$$\dot{u}^0 = \alpha\phi + \dot{\lambda}^0 v. \tag{2.19}$$

Proof. It follows from (2.8) that there is a vector w that satisfies

$$G_u^0 w = -G_\lambda^0. \tag{2.20}$$

In view of (2.8) and (2.12), the general solution of (2.20) is given by $w(\gamma) = w + \gamma\phi$ for $\gamma \in \mathbb{R}$. Assume first that $G_\lambda^0 \neq 0$. Then the vectors w and ϕ are linearly independent. Therefore, there is a unique constant γ, such that $v = w(\gamma)$ and ϕ are orthogonal. On the other hand, if $G_\lambda^0 = 0$, then Eqs. (2.6) and (2.18) imply that $v = 0$.

We turn to the proof of (2.19). Substitute $G_u^0 v = -G_\lambda^0$ into (2.10) for $s = s^0$. We obtain

$$0 = G_u^0 \dot{u}^0 + G_\lambda^0 \dot{\lambda}^0 = G_u^0 \dot{u}^0 - (G_u^0 v)\dot{\lambda}^0,$$

which in view of (2.12) shows (2.19). □

Assume that the conditions of Proposition 2.5 hold. Differentiate (2.10) with respect to s and evaluate the expression obtained at $s = s^0$. This yields

$$G_u^0 \ddot{u}^{\,0} + G_u^0 \ddot{\lambda}^0 + G_{uu}^0 \dot{u}^0 \dot{u}^0 + 2G_{u\lambda}^0 \dot{u}^0 \dot{\lambda}^0 + G_{\lambda\lambda}^0 \dot{\lambda}^0 \dot{\lambda}^0 = 0, \tag{2.21}$$

where $\ddot{u}^{\,0}$ and $\ddot{\lambda}^0$ denote second-order derivatives of $u(s)$ and $\lambda(s)$ with respect to s evaluated $s = s^0$. Multiply (2.21) by ψ^{T} from the left and recall that $\psi^{\mathrm{T}} G_u^0 = 0$ to obtain

$$\psi^{\mathrm{T}} G_{uu}^0 \dot{u}^0 \dot{u}^0 + 2\psi^{\mathrm{T}} G_{u\lambda}^0 \dot{u}^0 \dot{\lambda}^0 + \psi^{\mathrm{T}} G_{\lambda\lambda}^0 \dot{\lambda}^0 \dot{\lambda}^0 = 0.$$

Replacing $\dot{u}^0$ by the right-hand side of (2.19), $\dot{\lambda}^0$ by β, and letting

$$\begin{aligned} a_{11} &:= \psi^{\mathrm{T}} G_{uu}^0 \phi\phi, \\ a_{12} &:= \psi^{\mathrm{T}} G_{uu}^0 \phi v + \psi^{\mathrm{T}} G_{u\lambda}^0 \phi, \\ a_{22} &:= \psi^{\mathrm{T}} G_{uu}^0 vv + 2\psi^{\mathrm{T}} G_{u\lambda}^0 v + \psi^{\mathrm{T}} G_{\lambda\lambda}^0, \end{aligned}$$

yields

$$a_{11}\alpha^2 + a_{12}\alpha\beta + a_{22}\beta^2 = 0. \tag{2.22}$$

Eq. (2.22) is usually referred to as the algebraic bifurcation equation. It can be shown, see, e.g., [16, Section 5.20] that its discriminant $D := a_{12}^2 - 4a_{11}a_{22}$ is nonnegative. If $D > 0$, then (2.22) has two distinct roots, (α_1, β_1) and (α_2, β_2). Each root (α_*, β_*) corresponds to a smooth solution path for (1.1) in a neighborhood of the bifurcation point

$$\begin{aligned} u(s) &= u^0 + (s - s^0)[\alpha_*(s)v + \beta_*(s)\phi] + (s - s^0)^2 w_*(s), \\ \lambda(s) &= \lambda^0 + (s - s^0)\alpha_*(s), \end{aligned}$$

where

$$\psi^{\mathrm{T}} w_*(s) = 0, \quad \alpha_*(s^0) = \alpha_*, \quad \beta_*(s^0) = \beta_*.$$

The quantity $\psi^{\mathrm{T}} G_{uu}^0$, required for the evaluation of the coefficients a_{ij} in (2.22), is in general not available. Typically, approximations of $\psi^{\mathrm{T}} G_{uu}^0$ in terms of G_u are used in computations.

The perturbed bifurcation method avoids the computation of these approximants as well as the solution of the algebraic bifurcation Eq. (2.22). This method is based on the observation that the set of points in the domain of $G(u,\lambda)$ where the Jacobian $G_u(u,\lambda)$ is nonsingular are dense and the Jacobian is singular at bifurcation points. Therefore, if (u^0,λ^0) is not a regular point, then we can perturb the right-hand side in Eq. (1.1), i.e., we replace (1.1) by

$$G(u,\lambda)=\varepsilon q, \quad u=u(\lambda), \tag{2.23}$$

where $q\in\mathbb{R}^n$, $\|q\|=1$ and $\varepsilon\neq 0$, so that G_u is nonsingular at the solution of (2.23). Upon deletion of a small neighborhood containing in its interior the bifurcation point (u^0,λ^0), the two smooth solutions branches intersecting at the bifurcation point generate four different solution branches. It can be shown that the two solution branches which belong to the same path lie adjacent to each other, and therefore paths do not cross in the neighborhood of a bifurcation point. Hence, path following across a bifurcation point can be viewed as a limit case of following two regular paths. Further details on the perturbed bifurcation method are discussed by Georg [13] and Keller [16].

2.2. Computation

In this subsection we consider the quantities that need to be computed for path following. The algorithms of Section 4 for computing these quantities are based on the discussion of the present subsection. At a regular point (u^0,λ^0) of a solution path, we compute an initial approximate solution of (1.1) by Euler's method (2.3), where $\dot{u}_0$ is defined in (2.2). The vector $\dot{u}_0$ is determined by solving the linear system of equations (2.2). This system has a unique solution at any regular point on the solution path, because the Jacobian matrix G_u^0 is nonsingular there.

The iterates u_k of Newton's method are solutions of the linear systems (2.5). For each value of λ several systems (2.5) may have to be solved. It is important to monitor if any Jacobian matrix G_u^k determined during the iterations with Newton's method is singular, because a singular Jacobian may imply that (2.5) does not have a solution.

If G_u^k is singular and (u_k,λ) satisfies (1.1), then we are either at a bifurcation point or at a turning point. In either case we cannot use Newton's method to compute the next point on the path. However, if the eigenvector corresponding to the zero eigenvalue of the Jacobian matrix is available, then we can introduce a pseudo-arclength parameterization and use the bordering algorithm to find the next point on the path. The bordering algorithm requires that we solve linear systems of equations of the form (2.15). Conditions under which these linear systems have a unique solution are stated in the following proposition.

Proposition 2.6. *Let* $\tilde{G}^k$ *be defined by* (2.17). *The following situations can be distinguished*:

(i) *If* G_u^k *is nonsingular, then* $\tilde{G}^k$ *is nonsingular if and only if*

$$\dot{\lambda}^0-\dot{u}^{0\mathrm{T}}(G_u^k)^{-1}G_\lambda^k\neq 0. \tag{2.24}$$

(ii) *If* G_u^k *is singular and* $\dim\mathcal{N}(G_u^k)=1$, *then* $\tilde{G}^k$ *is nonsingular if and only if* $G_\lambda^k\notin\mathcal{R}(G_u^k)$ *and* $\dot{u}^0\notin\mathcal{R}((G_u^k)^{\mathrm{T}})$.

(iii) *If* G_u^k *is singular and* $\dim\mathcal{N}(G_u^k)>1$, *then* $\tilde{G}^k$ *is singular*.

Proof. The proposition follows by elementary matrix manipulations; see, e.g., [15] or [16, p. 76]. Note that the left-hand side of (2.24) is a Schur complement. Assume that both (u^0, λ^0) and (u^k, λ^k) lie on a solution path for (1.1). Then (2.24) expresses that the tangents $(\dot{u}^0, \dot{\lambda}^0)$ and $(\dot{u}^k, \dot{\lambda}^k)$ must not be orthogonal. □

Several solution methods for linear systems of equations of form (2.15) will be discussed. Different methods are tailored to matrices (2.17) with different properties.

We use notation (2.16). It is convenient to write (2.15) in the form

$$G_u^k \Delta u + G_\lambda^k \Delta\lambda = -G^k, \tag{2.25}$$

$$\dot{u}^{0\mathrm{T}} \Delta u + \dot{\lambda}^0 \Delta\lambda = -N^k. \tag{2.26}$$

Assume that G_u^k is nonsingular and $\dot{\lambda}^0 \neq 0$. We obtain from (2.26) that

$$\Delta\lambda = \frac{1}{\dot{\lambda}^0}(-N^k - \dot{u}^{0\mathrm{T}} \Delta u), \tag{2.27}$$

which, substituted into (2.25), yields

$$\left(G_u^k - \frac{1}{\dot{\lambda}^0}(G_\lambda^k \dot{u}^{0\mathrm{T}})\right) \Delta u = -G^k + \frac{1}{\dot{\lambda}^0} G_\lambda^k N^k. \tag{2.28}$$

Thus, when $\dot{\lambda}^0 \neq 0$, we can compute Δu by solving (2.28) and $\Delta\lambda$ from (2.27). We will refer to this as the *bordering algorithm for regular points*. The matrix in (2.28) is a rank-one modification of G_u^k. It is nonsingular if and only if (2.24) is satisfied, i.e., if and only if the system of equations (2.25)–(2.26) has a unique solution.

The bordering algorithm for a regular point described above cannot be used at a turning point (u^0, λ^0) since $\dot{\lambda}^0 = 0$ there. We now derive a simple solution method for system (2.25)–(2.26) for $k = 0$ under the assumption that (u^0, λ^0) is a turning point on the solution path. Note that the right-hand side of (2.25) vanishes for $k = 0$ because the turning point is on the solution path. Multiply (2.25) by ψ^{T} from the left and recall that $\psi^{\mathrm{T}} G_u^0 = 0$; see (2.12). We obtain

$$\psi^{\mathrm{T}} G_\lambda^0 \Delta\lambda = 0. \tag{2.29}$$

The factor $\psi^{\mathrm{T}} G_\lambda^0$ does not vanish because $\psi \in \mathcal{N}(G_u^{0\mathrm{T}})$ and it follows from (2.7) that $G_\lambda^0 \notin \mathcal{N}(G_u^{0\mathrm{T}})^\perp$. We conclude that $\Delta\lambda = 0$. Eq. (2.25) simplifies to

$$G_u^0 \Delta u = 0, \tag{2.30}$$

which is satisfied by

$$\Delta u = \alpha\phi, \quad \forall \alpha \in \mathbb{R}. \tag{2.31}$$

We determine α so that Δu satisfies (2.26), i.e.,

$$\alpha = -\frac{N^0}{\dot{u}^{0\mathrm{T}} \phi}. \tag{2.32}$$

The denominator in (2.32) does not vanish due to (2.11).

Having determined the corrections Δu and $\Delta\lambda = 0$ of the turning point (u^0, λ^0) as described above, yields the approximation $u_1 = u^0 + \Delta u$ and $\lambda_1 = \lambda^0$ of the solution of (2.13)–(2.14). The subsequent

Newton steps require the solution of linear systems (2.25)–(2.26) for $k = 1, 2, \ldots$. The Jacobian G_u^k of these systems is in general nonsingular. We outline how the system (2.25)–(2.26) can be solved when G_u^k is nonsingular and $\dot{\lambda}^0 = 0$. Compute the solutions y and z of the linear systems

$$G_u^k z = G_\lambda^k, \tag{2.33}$$

$$G_u^k y = -G^k. \tag{2.34}$$

Then

$$\Delta u = y - z\Delta\lambda \tag{2.35}$$

satisfies (2.25) for arbitrary $\Delta\lambda \in \mathbb{R}$. Substituting (2.35) into (2.26) gives

$$\Delta\lambda = \frac{N^k + \dot{u}^{0\mathrm{T}} y}{\dot{u}^{0\mathrm{T}} z}. \tag{2.36}$$

Thus, we solve (2.25)–(2.26) for $k = 0$ by using formulas (2.29)–(2.32), and for $k \geqslant 1$ by first solving the linear systems (2.33) and (2.34) and then using (2.35)–(2.36). We refer to this method for determining Δu and $\Delta\lambda$ at a turning point as the *bordering algorithm for simple turning points*.

The unit tangent vector $(\dot{u}^0, \dot{\lambda}^0)$ to the solution path at (u^0, λ^0) plays a central role in the bordering algorithm. We described how it can be computed. Eqs. (2.9) and (2.10) can be written as

$$G_u^0 \dot{u}^0 + G_\lambda^0 \dot{\lambda}^0 = 0, \tag{2.37}$$

$$\|\dot{u}^0\|^2 + |\dot{\lambda}^0|^2 = 1. \tag{2.38}$$

We first consider the case when G_u^0 is nonsingular and $\dot{\lambda}^0 \neq 0$. Then we can express $\dot{u}^0$ as

$$\dot{u}^0 = \dot{\lambda}^0 \chi, \tag{2.39}$$

where χ solves the linear system

$$G_u^0 \chi = -G_\lambda^0. \tag{2.40}$$

Eqs. (2.38) and (2.39) yield

$$\dot{\lambda}^0 = \frac{\pm 1}{\sqrt{1 + \|\chi\|^2}}. \tag{2.41}$$

The sign of $\dot{\lambda}^0$ is chosen so that the tangent vector points in the positive direction of the path. If $(\dot{u}^{-1}, \dot{\lambda}^{-1})$ is the unit tangent vector at a previous point on the path, we choose the sign in the right-hand side of (2.41) so that the cosine of the angle between $(\dot{u}^0, \dot{\lambda}^0)$ and $(\dot{u}^{-1}, \dot{\lambda}^{-1})$ is positive, i.e.,

$$\dot{u}^{-1}\dot{u}^0 + \dot{\lambda}^{-1}\dot{\lambda}^0 > 0 \tag{2.42}$$

or, equivalently,

$$\dot{\lambda}^0(\dot{u}^{-1}\chi + \dot{\lambda}^{-1}) > 0.$$

If the wrong orientation of the tangent vector is chosen, then the bordering algorithm will backtrack the path already computed.

We turn to the solution on (2.37)–(2.38) at a turning point. Then G_u^0 is singular, $\dot{\lambda} = 0$ and $\dot{u}^0 \in \mathcal{N}(G_u^0)$. The computation of the unit tangent vector at a turning point amounts to computing the eigenvector of Euclidean norm one of the Jacobian associated with the zero eigenvalue and choosing its orientation according to (2.42).

We conclude this section with a discussion of the solution of linear systems with a nonsingular matrix (2.17), whose $n \times n$ leading principal submatrix G_u^k is singular with $\dim \mathcal{N}(G_u^k) = 1$. The right-hand side is a general vector. Thus, consider the linear system

$$G_u^k x + G_\lambda^k \xi = g, \tag{2.43}$$

$$\dot{u}^{0\mathrm{T}} x + \dot{\lambda}^0 \xi = \gamma \tag{2.44}$$

and let the vectors ϕ and ψ satisfy (2.12). Then the solution of (2.43)–(2.44) can be expressed as

$$x = y - \xi z + \alpha\phi, \tag{2.45}$$

$$\xi = \frac{\psi^{\mathrm{T}} g}{\psi^{\mathrm{T}} G_\lambda^k}, \tag{2.46}$$

where

$$G_u^k y = g - (\psi^{\mathrm{T}} g)\psi, \quad G_u^k z = G_\lambda^k - (\psi^{\mathrm{T}} G_\lambda^k)\psi\,, \tag{2.47}$$

$$\phi^{\mathrm{T}} y = \phi^{\mathrm{T}} z = 0, \tag{2.48}$$

$$\alpha = \frac{\gamma - \dot{\lambda}^0 \xi - \dot{u}^{0\mathrm{T}}(y - \xi z)}{\dot{u}^{0\mathrm{T}} \phi}. \tag{2.49}$$

This can be seen as follows. Multiplying Eq. (2.43) by ψ^{T} yields (2.46). The denominator is nonvanishing because $\psi \in \mathcal{R}(G_u^k)^\perp$ and, by Proposition 2.6, $G_\lambda^k \notin \mathcal{R}(G_u^k)$. The linear systems of equations (2.47) are consistent and the orthogonality conditions (2.48) determine the solutions y and z of these systems uniquely. Eq. (2.43) is satisfied by x and ξ given by (2.45)–(2.46) for any $\alpha \in \mathbb{R}$. Eq. (2.49) determines α so that Eq. (2.44) is satisfied. It follows from Proposition 2.6 that the denominator in (2.49) does not vanish.

3. The IRBL method

We outline the implicitly restarted block-Lanczos (IRBL) method for the computation of a few of the smallest eigenvalues and associated eigenvectors of a symmetric matrix $A \in \mathbb{R}^{n\times n}$. This method is used in the algorithms for path following described in Section 4. In the applications discussed there, A is a Jacobian matrix G_u associated with the nonlinear system of equations (1.1). The IRBL method helps us determine whether the Jacobian matrix is singular. For singular matrices, the method yields a basis of the null space. It is important to detect singular points on the solution path during path following, and knowledge of the null space of singular Jacobian matrices helps us to follow the path across a singular point.

The IRBL method is an iterative scheme for the computation of a few eigenvalues and associated eigenvectors in a specified portion of the spectrum. It is based on the recurrence relations of the

block-Lanczos algorithm. The IRBL method was introduced in [4]; here we only discuss aspects pertinent for the application considered in Section 4. When the block-size r is chosen to be one, the IRBL method simplifies to the implicitly Restarted Lanczos (IRL) method described in [2,6]. In our experience the IRBL method is better suited for computing eigenvalue–eigenvector pairs associated with multiple or close eigenvalues than the IRL method.

In the present paper, we choose the block-size r to be the number of desired eigenpairs. Let $\{v_j\}_{j=1}^r$ be a set of orthonormal vectors in $\mathbb{R}^n$ and introduce the matrix $V_r = [v_1, v_2, \ldots, v_r]$. Assume for simplicity that the block-Lanczos process does not break down. Then m steps the block-Lanczos process yield a symmetric block-tridiagonal matrix $T_{mr} \in \mathbb{R}^{mr\times mr}$ with $r\times r$ blocks and upper triangular subdiagonal blocks, such that

$$AV_{mr} = V_{mr}T_{mr} + F_rE_r^{\mathrm{T}}, \tag{3.1}$$

where $V_{mr} \in \mathbb{R}^{n\times mr}$, $V_{mr}I_{mr\times r} = V_r$, $V_{mr}^{\mathrm{T}}V_{mr} = I_{mr}$ and $F_r \in \mathbb{R}^{n\times r}$ satisfies $V_{mr}^{\mathrm{T}}F_r = 0$. Here I_{mr} denotes the $mr\times mr$ identity matrix, $I_{mr\times r} \in \mathbb{R}^{mr\times r}$ consists of the first r columns of I_{mr} and E_r consists of the r last columns of I_{mr}. The columns of V_{mr} span the Krylov subspace $\mathscr{K}_{mr}(A, V_r) := \mathrm{span}\{V_r, AV_r, \ldots, A^{m-1}V_r\}$ and

$$T_{mr} = V_{mr}^{\mathrm{T}}AV_{mr}.$$

Introduce the spectral factorization

$$T_{mr} = Y_{mr}\Theta_{mr}Y_{mr}^{\mathrm{T}},$$

where $\Theta_{mr} = \mathrm{diag}[\theta_1, \theta_2, \ldots, \theta_{mr}]$, $Y_{mr} \in \mathbb{R}^{mr\times mr}$, $Y_{mr}^{\mathrm{T}}Y_{mr} = I_{mr}$. The eigenvalues $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_{mr}$ of T_{mr} approximate eigenvalues of A and are usually referred to as Ritz values. The vectors $x_j = V_{mr}y_j$, $1 \leqslant j \leqslant mr$, approximate eigenvectors of A and are referred to as Ritz vectors. It follows from (3.1) that the residual error $Ax_j - x_j\theta_j$ associated with the Ritz pair (θ_j, x_j) satisfies

$$\|Ax_j - x_j\theta_j\| = \|(AV_{mr} - V_{mr}T_{mr})y_j\| = \|F_rE_r^{\mathrm{T}}y_j\|.$$

We say that a Ritz pair (θ_j, x_j) is an acceptable approximation of an eigenpair of the matrix A if

$$\|F_rE_r^{\mathrm{T}}y_j\| \leqslant \varepsilon, \tag{3.2}$$

where $\varepsilon > 0$ is a small constant. Then (θ_j, x_j) is an eigenpair of a matrix $\hat{A} \in \mathbb{R}^{n\times n}$, such that $\|A - \hat{A}\| \leqslant \varepsilon$.

As the value of m increases the Ritz pairs become better approximations of eigenpairs of A. On the other hand each unit increase in m requires that r additional n-vectors be stored. Therefore, when the matrix A is very large and a large number of Lanczos steps are needed, use of secondary computer memory may become necessary. To avoid the slow-down which typically occurs when using secondary storage, a restarting scheme is employed. The recurrence relation of the IRBL method described in [4] for restarting allows us to compute

$$\hat{V}_r = \psi_m(A)V_rR \tag{3.3}$$

from the block-Lanczos decomposition (3.1) without additional evaluations of matrix–vector products with the matrix A. Here $\psi_m(t)$ is a polynomial of degree m, R is an upper triangular matrix and $\hat{V}_r$ is an $n \times r$ matrix with orthonormal columns which will be used as the initial block for the next block-Lanczos recursion. We seek to choose the polynomials ψ_m so that the columns of $\hat{V}_r$ are in, or close to, the invariant subspace of A associated with the desired eigenvalues. To achieve this, we

allocate the zeros $z_1, z_2, \ldots, z_m$ of ψ_m in one or two intervals away from the wanted portion of the spectrum. For example, if the smallest eigenvalues of A are desired, then we choose the z_j to be in the interval $[\theta_{m(r-1)}, \theta_{mr}]$. When we wish to determine a few nearby nonextreme eigenvalues, the zeros are allocated in two interval, one on each side of the set of desired eigenvalues. Details on how to select the zeros can be found in [2–4].

Once the matrix $\hat{V}_r$ and its columns have been orthogonalized against each other as well as against any converged eigenvectors, we compute a new block-Lanczos factorization (3.1) with $V_r := \hat{V}_r$. We repeat this process until r eigenpairs have been determined. The reliability of this scheme is illustrated by computed examples reported in [2–4].

We conclude this section with a few implementation issues. In each application of the IRBL method we determine the initial matrix V_r by orthogonalizing r columns of random numbers from the standard normal distribution; see Ericsson and Ruhe [11] for an explanation of the advantage of using normally distributed instead of uniformly distributed random numbers. Further, we note that in order to reduce data movement, it may be advantageous to implement the block-Lanczos process so that the r vectors in a block are multiplied by the matrix A simultaneously.

Columns of the matrix V_{mr} that are associated with the same block of r vectors are generated by first applying a three-term recursion formula determined by (3.1) followed by orthogonalization of the columns in the same block. In exact arithmetic and in the absence of break-down, these computations give a matrix V_{mr} with orthogonal columns. In order to secure orthogonality in the presence of round-off errors, we explicitly reorthogonalize the generated columns to all already available columns of the matrix. If the new columns generated fail to be numerically orthogonal after one reorthogonalization, then this signals that they are linearly dependent and a break down of the block-Lanczos process has occurred. Details of how to handle break downs will be discussed elsewhere. Here it suffices to say that break downs are easy to handle, and a procedure for this has been implemented in the code used for the numerical experiments reported in Section 5.

4. Algorithms for path following

We present several iterative methods for computing a solution path for nonlinear systems of equations (1.1) with a symmetric Jacobian G_u. First we describe the iterative method proposed in [5] for the simultaneous computation of a few eigenpairs associated with the eigenvalues of smallest magnitude and the solution of a linear system.

Consider the nonlinear system of equations (1.1) and assume that (u^0, λ^0) is a regular point on a solution path. We would like to determine the point (u^1, λ^1) on the path, where $u^1 = u(\lambda^1)$ and $\lambda^1 := \lambda^0 + \Delta\lambda$. The application of the Euler–Newton continuation method requires the solution of a sequence of linear systems of equations of the form (2.5) with $\lambda = \lambda^1$. Assume that the iterates u_k defined by (2.4) converge to u^1 as k increases. If $G_u(u^1, \lambda^1)$ is singular, then by a continuity argument the matrices $G_u(u_k, \lambda^1)$ for k sufficiently large have an eigenvalue close to the origin. Therefore, by tracking the eigenvalues of smallest magnitude of the Jacobian matrices $G_u(u_k, \lambda^1)$ while computing iterates (2.4), we can detect when we approach a singular point on the solution path. Algorithm 4.1 below determines an approximate solution of (2.5) while simultaneously computing r Ritz pairs $\{(\theta_\ell^k, x_\ell^k)\}_{\ell=1}^r$ that approximate eigenpairs of $G_u(u_k, \lambda^1)$ associated with the r smallest eigenvalues.

Algorithm 4.1. Simultaneous solution of linear system and eigenproblem:
Input: λ^1, u_0, $A := G_u(u_k, \lambda^1)$, $b := -G(u_k, \lambda^1), m, r, V_r, \varepsilon$.
Output: Approximations $\{\theta_\ell^k\}_{\ell=1}^r$ *of the* r *smallest eigenvalues of* $G_u(u_k, \lambda^1)$, *approximations* $\{x_\ell^k\}_{\ell=1}^r$ *of associated orthonormal approximate eigenvectors and an approximate solution* $\Delta\tilde{u}$ *of* (2.5) *with* $\lambda = \lambda^1$.

1. $\Delta\tilde{u} := 0$.
2. *for* $\mu := 1, 2, \ldots$ *until* r *approximate eigenpairs found*
3. *Orthogonalize* V_r *against already computed approximate eigenvectors.*
4. *Compute block-Lanczos decomposition* (3.1) *with initial block* V_r.
5. *Update approximate solution of* (2.5):
   Solve $T_{mr}\Delta y = V_{mr}^{\mathrm{T}} b$, $\Delta\tilde{u} := \Delta\tilde{u} + V_{mr}\Delta y$.
6. *Determine Ritz pairs that satisfy* (3.2) *and store the Ritz vectors. We refer to the stored vectors as approximate eigenvectors and denote them by* $u_\ell^{(k)}$.
7. *Apply* m *shifts* $\{z_j\}_{j=1}^m$ *to determine the matrix* $\hat{V}_r$ *given by* (3.3). *The* z_j *are chosen to be fast Leja shifts described in* [3]. *Let* $V_r := \hat{V}_r$.
8. *endfor*
9. *Improve approximate solution* $\Delta\tilde{u}$ *by a conjugate gradient method. Terminate the iterations when the residual error is of norm smaller than* ε. *Denote the computed solution by* $\Delta\tilde{u}$.

Step 6 of Algorithm 4.1 yields approximations of the r eigenpairs associates with the eigenvalues of smallest magnitude. The solution of (2.5) can be expanded in terms of eigenvectors of the matrix. Step 5 essentially removes eigenvector components associated with the smallest eigenvalues from this expansion. The solution of the linear system that is solved in Step 9 can be expressed in terms of an expansion of eigenvectors associated with the remaining (larger) eigenvalues. Therefore, Step 5 of Algorithm 4.1 can be thought of as preconditioning the linear system (2.5).

In our numerical examples in Section 5, we use the conjugate gradient method designed for the solution of inconsistent linear systems with a symmetric possibly indefinite matrix described in [7].

The matrix $V_r \in \mathbb{R}^{n\times r}$ required as input for Algorithm 4.1 is assumed to have orthonormal columns. In the very first application of Algorithm 4.1, V_r is initialized with normally distributed random entries in each column, which then are orthonormalized. In later applications of Algorithm 4.1 the columns of the input matrix V_r are chosen to be the approximate eigenvectors $\{w_\ell^k\}_{\ell=1}^r$ determined during the most recent application of the algorithm.

Assume now that G_u^0 is singular with $\dim \mathcal{N}(G_u^0) = 1$, and let ϕ satisfy (2.12). It follows from the symmetry of the Jacobian and a discussion analogous to the one following (2.29) that the point (u^0, λ^0) is a turning point on the solution path if and only if $\phi^{\mathrm{T}} G_\lambda^0 \neq 0$. Let (u^0, λ^0) be a turning point. We describe how to determine the pseudo-arclength parameterization required by the bordering algorithm. This parameterization requires that the unit tangent vector

$$(\dot{u}^0 \dot{\lambda}^0) = (\pm\phi, 0) \tag{4.1}$$

be available. If the unit tangent vector $(\dot{u}^{-1}, \dot{\lambda}^{-1})$ at the previous point on the path is available, then we choose the sign in the right-hand side of (4.1) so that (2.42) is satisfied. When the tangent vector $(\dot{u}^{-1}, \dot{\lambda}^{-1})$ is not known, we approximate it by the secant $(u^0, \lambda^0) - (u^{-1}, \lambda^{-1})$. The solution of (2.25)–(2.26) for $k = 0$ is given by $\Delta\lambda = 0$ and (2.31)–(2.32).

The performance of the Euler–Newton predictor–corrector method (2.3)–(2.5) with the natural parameter λ requires that the steps $\Delta\lambda$ be small in the vicinity of a turning point. Therefore, we switch from the natural to the pseudo-arclength parameterization when we are close to a turning point and compute the next point on the path via the following variant of the bordering algorithm. We first solve the linear systems of equations (2.33) and (2.34) and then determine $\Delta\lambda$ from

$$\Delta\lambda = \frac{\dot{u}^{0\mathrm{T}} y + N^k}{\dot{u}^{0\mathrm{T}} z - \dot{\lambda}^0} \tag{4.2}$$

and Δu from (2.35). The vector $(\Delta u, \Delta\lambda)$ satisfies (2.25)–(2.26). Formula (4.2) simplifies to (2.36) when $\dot{\lambda}^0 = 0$. Since in finite precision arithmetic $|\dot{\lambda}^0|$ is small but possibly nonvanishing at a computed approximate turning point, we use (4.2) instead of (2.36) in computations. We switch from natural to pseudo-arclength parameterization when either the Jacobian has an eigenvalue of magnitude smaller than a tolerance ε_s or the step size required with the natural parameterization is below a given threshold. The following algorithm summarizes how to organize the calculation to compute regular points on a solution path across a turning point.

Algorithm 4.2. Path following around a turning point:
Input: λ^0, u^0, $\dot{\lambda}^0$, $\dot{u}^0$, $G_u(u^0, \lambda^0)$, Δs, $\varepsilon, k_{\max}$.
Output: u^1, λ^1.

```
convergence := false, u_0 := u^0, λ_0 := λ^0.
while not convergence,
   for k := 0, 1, ..., k_max
      Solve (2.33) for z and (2.34) for y.
      Compute Δλ from (4.2).
      Compute Δu from (2.35).
      u_{k+1} := u_k + Δu, λ_{k+1} := λ_k + Δλ.
      if ||G(u_{k+1}, λ_{k+1})|| < ε then
         convergence := true, u^1 := u_{k+1}, λ^1 := λ_{k+1}, exit.
      endif
   endfor
   Δs := Δs/2.
endwhile
```

We turn to the problem of branch switching at a bifurcation point. In finite precision arithmetic computation, we may not be able to determine exact bifurcation points. Instead we are in a position to compute approximations of bifurcation points. We refer to these points as perturbed bifurcation points. The presence of a perturbed bifurcation point on the solution path is signaled numerically by the Jacobian matrix becoming nearly singular and by the magnitude of $\phi^{\mathrm{T}} G_\lambda^0$ being very small.

Let (u^0, λ^0) be a perturbed bifurcation point on the solution path. We discuss how to switch path there. In view of (2.10) the tangent vectors at the bifurcation point are in the null space of the

matrix $[G_u^0, G_\lambda^0]$ and can, by Proposition 2.5, be written as

$$\tau := \alpha \begin{bmatrix} \phi \\ 0 \end{bmatrix} + \beta \begin{bmatrix} v \\ 1 \end{bmatrix}.$$

A tangent to the active branch at (u^0, λ^0) is determined by $\beta = \dot{\lambda}^0$ and $\dot{u}^0 = \alpha\phi + \beta v$, where the coefficient α easily can be found by using the orthogonality (2.18). We search for the other branch by moving in a direction parallel to τ starting from a point at a distance ε_b from (u^0, λ^0) on the normal to τ in the plane $\mathcal{N}([G_u^0, G_\lambda^0])$. Note that the vector

$$\hat{\tau} := \hat{\alpha} \begin{bmatrix} \phi \\ 0 \end{bmatrix} + \hat{\beta} \begin{bmatrix} v \\ 1 \end{bmatrix}$$

with $\hat{\alpha} = t\beta(1 + \|v\|^2)$ and $\hat{\beta} = -t\alpha\|\phi\|^2$ is orthogonal to τ and in $\mathcal{N}([G_u^0, G_\lambda^0])$ for any $t \geqslant 0$. We choose the scaling factor t so that $\hat{\tau}$ is of unit length. Thus, we would like to determine a point $(u^2, \lambda^2) \in \mathbb{R}^{n+1}$ with coordinates of the form

$$\begin{aligned} u^2 &= u^0 + \varepsilon_b(\hat{\beta}v + \hat{\alpha}\phi) + w, \\ \lambda^2 &= \lambda^0 + \varepsilon_b\hat{\beta} + \zeta, \end{aligned} \tag{4.3}$$

where $w \in \mathbb{R}^n$ and $\zeta \in \mathbb{R}$ are chosen so that

$$G(u^2, \lambda^2) = 0, \tag{4.4}$$

$$N(u^2, \lambda^2) = 0 \tag{4.5}$$

and $N(u^2, \lambda^2) := (\hat{\beta}v^{\mathrm{T}} + \hat{\alpha}\phi^{\mathrm{T}})w + \hat{\beta}\zeta$. Condition (4.5) imposes that the vector (w, ζ) is orthogonal to $\hat{\tau}$. We compute w and ζ by applying Newton's method to the system (4.4)–(4.5) with initial approximate solution $w = 0$ and $\zeta = 0$. The sequence of linear systems of equations obtained in this manner are solved by the methods described in Section 2.

The computations for branch switching at a bifurcation point (u^0, λ^0) are summarized by the following algorithm. The vector ϕ is defined by (2.12).

Algorithm 4.3. Branch switching at a bifurcation point:
Input: λ^0, $u^0, \dot{\lambda}^0, \dot{u}^0, \phi, \varepsilon_b$.
Output: u^2, λ^2.
1. $\alpha := \phi^{\mathrm{T}}\dot{u}^0$, $\beta := \dot{\lambda}^0$.
2. $v := (\dot{u}^0 - \alpha\phi)/\beta$.
3. $\hat{\alpha} := t\beta(1 + \|v\|^2)$, $\hat{\beta} := -t\alpha\|\phi\|^2$ *with* $t > 0$ *chosen so that* $\hat{\alpha}^2 + \hat{\beta}^2 = 1$.
4. *Solve* (4.4)–(4.5) *for* (u^2, λ^2) *given by* (4.3) *using Newton's method.*

We conclude this section with an algorithm that outlines how to compute a new point on a solution path for (1.1) with special handling of turning and bifurcation points.

Algorithm 4.4. Iterative method for path following:
Input: λ^0, λ^1, u^0, $\varepsilon, \varepsilon_n, \varepsilon_s, \theta_{\min} = eigenvector\ of\ smallest\ magnitude\ of\ G_u^0$.
Output: u^1.

1. *if* $|\theta_{\min}| \geqslant \varepsilon_s$ *then*
   % (u^0, λ^0) *is a regular point.*
2. *Compute the Euler predictor* u_0.
3. *Use Newton's method to find* u^1, *such that* $\|G(u^1, \lambda^1\| < \varepsilon$. *Compute Newton iterates by Algorithm* 4.1.
4. *else*
   % *Test whether we are at a turning point or at a bifurcation point.*
5. *if* $|\phi^{\mathrm{T}} G_\lambda^0| > \varepsilon_n$ *then*
   % *Turning point*
6. *Compute* (u^1, λ^1) *by Algorithm* 4.2.
7. *else*
   % *Bifurcation point: first continue on active branch then switch branch.*
8. *Compute* (u^1, λ^1) *by Algorithm* 4.2.
9. *Switch branch by Algorithm* 4.3. *Compute points on other branch.*
10. *endif*
11. *endif*

5. Numerical examples

This section presents computations with the algorithms described in Section 4. The algorithms were implemented in MATLAB on a Silicon Graphics workstation. It is the purpose of the examples to illustrate the ease of use of the algorithms for path following in an interactive computing environment.

We report the number of times the Jacobian is used to multiply one or several n-vectors as a measure of the computational effort required. However, our algorithms have not (yet) been tuned for efficiency, and we expect that the computational effort required by carefully designed algorithms to be smaller. We return to this issue after the examples.

Example 5.1. Consider the nonlinear eigenvalue problem

$$\begin{aligned} -\mathscr{L}U - \lambda \sin(U) &= 0 \quad \text{in } \Omega, \\ U &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{5.1}$$

where $\Omega := \{(s,t)\colon 0 \leqslant s \leqslant 1,\ 0 \leqslant t \leqslant 1\}$ and $\partial\Omega$ denotes the boundary of Ω. We discretize Ω by a uniform grid with grid points $s_j := (j-1)h$ and $t_j := (j-1)h$ for $1 \leqslant j \leqslant \ell + 1$ and $h = 1/\ell$. The Laplacian $\mathscr{L}$ is approximated by the standard five-point stencil. This yields a system of $n := (\ell - 1)^2$ nonlinear algebraic equations of the form (1.1). The entries of the solution $u \in \mathbb{R}^n$ of (1.1) approximate $U(s,t)$ at the nodes s_j and t_j.

The discretization error is $\mathcal{O}(h^2)$ and we let $\varepsilon := h^2$ in the algorithms of Section 4. In particular, a Jacobian G_u is considered singular when it has an eigenvalue of magnitude smaller than h^2. Iterations

Table 1
Legend for Examples 5.1 and 5.2

EP	Computation of eigenpairs of Jacobian by the IRBL algorithm.
NW	Newton's method for solution of (1.1), eigenpairs of Jacobian by Algorithm 4.1.
EN	Continuation by the Euler–Newton method, step length $\Delta\lambda$.
SB	Switch branch using Algorithm 4.3, step length ε_b.
BR	Continuation by bordering using Algorithm 4.2, Jacobian regular, step length Δs.
BS	Continuation by bordering using Algorithm 4.2, Jacobian singular, step length Δs.

Table 2
Example 5.1: Solution path with bifurcation point

Step	Comput.	λ	Step length	Smallest eig. val.	Solut. norm	Matrix acces.	Mat.–vec. prod.	CG iter.
a	EP	18.0000	—	1.6987	0	58	113	0
b	EN	19.6987	1.6987	$-4 \cdot 10^{-5}$	0	72	137	0
c	SB	19.6987*	0.1	$5 \cdot 10^{-4}$	$1 \cdot 10^{-1}$	139	214	53
d	BS	19.6987	0.1	$2 \cdot 10^{-3}$	$2 \cdot 10^{-1}$	200	285	100
e	BR	19.7003*	0.1	$5 \cdot 10^{-3}$	$3 \cdot 10^{-1}$	223	318	109
f	BR	19.7016	0.1	$6 \cdot 10^{-3}$	$4 \cdot 10^{-1}$	256	371	115
g	BR	19.7136*	0.1	$3 \cdot 10^{-2}$	$9 \cdot 10^{-1}$	288	423	120
h	EN	19.8136*	0.1	$3 \cdot 10^{-1}$	$2 \cdot 10^{0}$	336	511	128
i	EN	20.3136*	0.5	$1 \cdot 10^{0}$	$6 \cdot 10^{0}$	434	679	138

with Newton's method are terminated as soon as an iterate has been found that gives a value of G of norm less than h^2. In the present example $\ell = 20$, $h = 5 \cdot 10^{-2}$ and $\varepsilon = 2.5 \cdot 10^{-3}$.

We use the algorithms of Section 4 to determine a bifurcation point and switch path in an interactive computing environment. The computations carried out are shown in the column "comput." of Table 2. The abbreviations used are explained in Table 1.

The boundary value problem (5.1) has the trivial solution $U(s,t) = 0$ for any value of λ, and the discretized problem has the solution $u = 0$. The Jacobian matrices associated with the solution $u = 0$ are singular when λ is an eigenvalue of the discretized Laplacian. We choose $(u, \lambda) = (0, 18)$ as initial point on the solution path and use the IRBL algorithm with block-size $r = 2$ and $m = 5$ block-Lanczos steps between the restarts to compute the two smallest eigenvalues and associated eigenvectors of the Jacobian matrix. This is Step (a) in Table 2. The table shows the computed approximation of the eigenvalue of the Jacobian closest to the origin in the column labeled "Smallest eig. val." In all computations reported the eigenvalue closest to the origin is also the smallest one. Two measures of the computational effort required for the computation of the eigenpairs are reported in the columns "Matrix acces." and "Mat.–vec. prod." The former column reports the number of matrix accesses, i.e., the number of times the Jacobian is multiplied by a vector or a block of r vectors. This count is of interest when the entries of the Jacobian are evaluated every time they are used in order to reduce the computer storage required, and measures the number of times each matrix entry has to be computed. This approach is used in the code for liquid crystal modeling described in [4] in order to reduce the storage requirement for the Jacobian, as well as in finite element codes for large-scale problems. The column "Mat.–vec. prod." shows the number of times the Jacobian matrix is multiplied by an

n-vector. Multiplying a block of r n-vectors by the Jacobian counts as r matrix–vector products. Whether this count is relevant depends on the size of the problem and the on architecture of the computer used. For large-scale problems, this count, in general, is of less interest than the number of matrix accesses.

We add the computed approximation 1.6987 of the smallest eigenvalue of the Jacobian determined in Step (a) to λ in order to obtain a nearly singular Jacobian matrix and take a step with the Euler–Newton method with $\Delta\lambda = 1.6987$. This is Step (b) in Table 2. The table reports the cumulative number of matrix accesses and matrix–vector products required. Thus, the computations for Step (b) require 16 matrix accesses and the evaluation of 24 matrix–vector products. We do not use the fact that the eigenvectors for the Jacobian at $\lambda = 18$ and 19.6987 are the same. The matrix–vector products reported are used to take a step with the Euler–Newton method and to verify that r eigenpairs and a solution u for $\lambda = 19.6987$ have been determined to desired accuracy.

The smallest eigenvalue of the Jacobian determined in this manner is in magnitude smaller than $\varepsilon = h^2$. We therefore consider the Jacobian singular. Algorithm 4.3 with $\varepsilon_b = 0.1$ is applied to determine a nontrivial solution u of (1.1) for λ close to 19.6987. Note that if u solves (1.1), then so does $-u$. The arithmetic work required is reported in step (c) in Table 2. Algorithm 4.1 is used with $m = 5$ block-Lanczos steps between restarts. The column "CG iter." displays the number of iterations, 53, carried out with the conjugate gradient method in Step 9 of Algorithm 4.1. These iterations are included in the count of matrix accesses and matrix–vector products reported in the table. Subsequent entries of the column "CG iter." report the cumulative number of iterations. The column "Solut. norm" of Table 2 displays the Euclidean norm of the computed solution u. Starred parameter values in the column "λ" indicate that the solution is plotted in Fig. 1. The (s,t)-plane in Figs. 1(c)–(i) displays the computed approximations of $U(s_j, t_k)$ as a function of j and k.

We turn to the computations of step (d). The Jacobian obtained in step (c) has an eigenvalue of magnitude smaller than h^2. We therefore consider the Jacobian singular and apply a bordering method with $\Delta s = 0.1$ to determine a new point on the solution path. Steps (e)–(g) in Table 2 differ from step (d) only in that a bordering method, Algorithm 4.2, for nonsingular Jacobian matrices is used.

Finally, we determine two points on the solution path with the Euler–Newton method using step lengths $\Delta\lambda = 0.1$ and 0.5, respectively. The computational work required is reported in Steps (h) and (i) in Table 2.

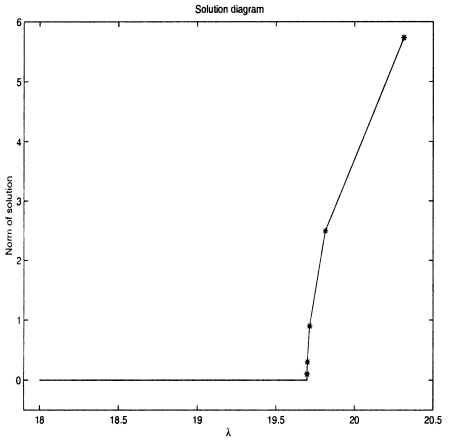
Example 5.2. The nonlinear boundary value problem

$$-\mathscr{L}U - \lambda \exp(U) = 0 \quad \text{in } \Omega,$$

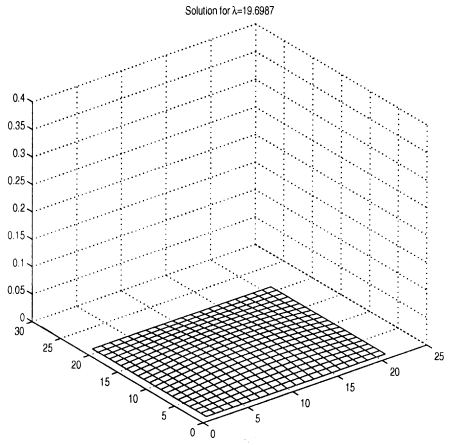
$$U = 0 \quad \text{on } \partial\Omega, \tag{5.2}$$

where Ω and $\partial\Omega$ are the same as in (5.1) is known as the Bratu problem and is a common test problem for path following methods. We discretize (5.2) in the same manner as Eqs. (5.1) of Example 5.1.

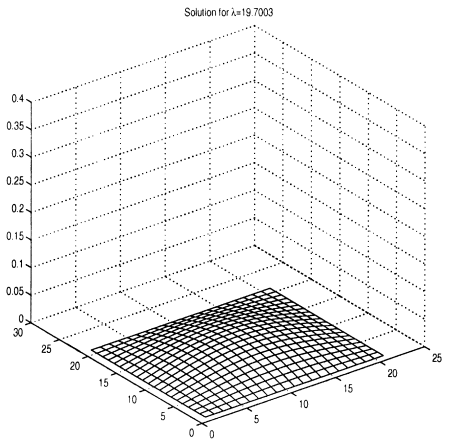
This example illustrates the application of the algorithms of Section 4 to traverse a turning point. The computational steps are displayed in Table 3, which is analogous to Table 2. We let the initial approximate solution of the nonlinear system of equations (1.1) determined by (5.2) with $\lambda = 5$ be



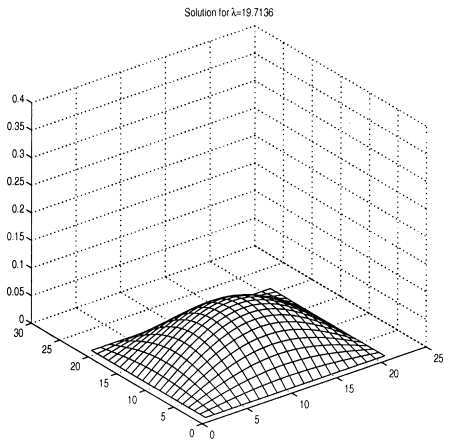
(p)



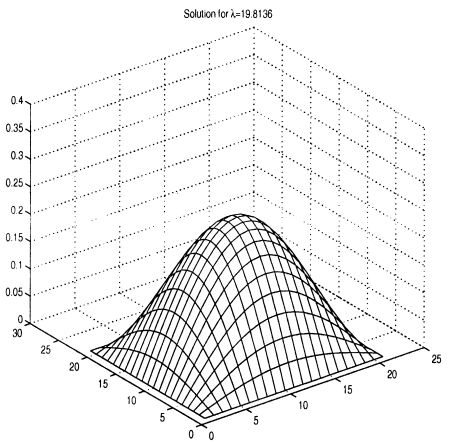
(c)



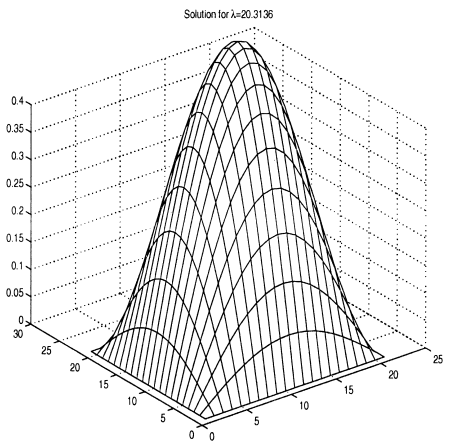
(e)



(g)



(h)



(i)

Fig. 1. Example 5.1: Solution path with bifurcation point. Figure (p) shows a solution path. Figures (c)–(i) display the solutions u associated with points on the graph in figure (p).

Table 3
Example 5.2: Solution path with turning point

Step	Comput.	λ	Step length	Smallest eig. val.	Solut. norm	Matrix acces.	Mat.–vec. prod.	CG iter.
a	NW	5.0000*	—	$1 \cdot 10^{1}$	$6.14 \cdot 10^{0}$	307	507	98
b	EN	6.0000	1.00	$8 \cdot 10^{0}$	$8.63 \cdot 10^{0}$	399	639	144
c	EN	6.5000	0.50	$5 \cdot 10^{0}$	$1.08 \cdot 10^{1}$	481	761	180
d	EN	6.7500*	0.25	$1 \cdot 10^{0}$	$1.33 \cdot 10^{1}$	581	916	217
e	BR	6.7655	0.75	$2 \cdot 10^{-1}$	$1.41 \cdot 10^{1}$	689	1064	278
f	BR	6.7658	0.10	$4 \cdot 10^{-2}$	$1.42 \cdot 10^{1}$	786	1191	338
g	BR	6.7658*	0.05	$-4 \cdot 10^{-2}$	$1.42 \cdot 10^{1}$	889	1319	409
h	BR	6.7655	0.10	$-2 \cdot 10^{-1}$	$1.43 \cdot 10^{1}$	991	1451	474
i	BR	6.7509	0.75	$-1 \cdot 10^{0}$	$1.51 \cdot 10^{1}$	1107	1602	548
j	EN	6.5009*	−0.25	$-7 \cdot 10^{0}$	$1.81 \cdot 10^{1}$	1259	1839	605
k	EN	5.5009*	−1.00	$-2 \cdot 10^{1}$	$2.36 \cdot 10^{1}$	1398	2058	656
l	EN	4.5009	−1.00	$-3 \cdot 10^{1}$	$2.78 \cdot 10^{1}$	1510	2225	707
m	EN	3.5009*	−1.00	$-5 \cdot 10^{1}$	$3.19 \cdot 10^{1}$	1622	2397	753
n	EN	2.5009	−1.00	$-8 \cdot 10^{1}$	$3.61 \cdot 10^{1}$	1742	2577	807
o	EN	1.5009	−1.00	$-1 \cdot 10^{2}$	$4.07 \cdot 10^{1}$	1927	2862	884

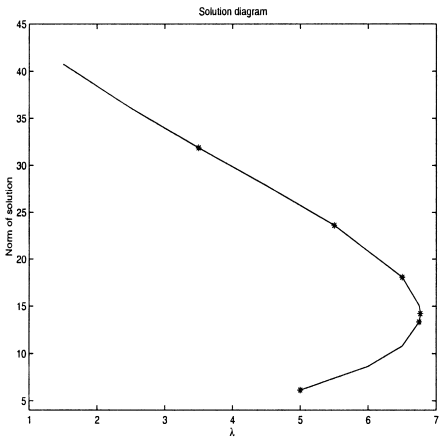


Fig. 2. Example 5.2: Solution path with turning point.

a random vector, and use Newton's method (2.4)–(2.5) to compute an approximate solution u of desired accuracy. The linear systems of Eqs. (2.5) in Newton's method are solved by Algorithm 4.1, which also gives eigenpairs of the Jacobian associated with the two smallest eigenvalues. The computational effort required is reported in Step (a) of Table 3. Steps (b)–(o) report path following by the Euler–Newton method and by bordering. All eigenvalues of all Jacobian matrices generated are of magnitude larger than $\varepsilon = h^2$. The Jacobian matrices are therefore considered nonsingular. Similarly as in Example 5.1, the computational effort reported is cumulative. For instance, the matrix accesses reported in Step (o) are for all computations in Steps (a)–(o). The matrix accesses and

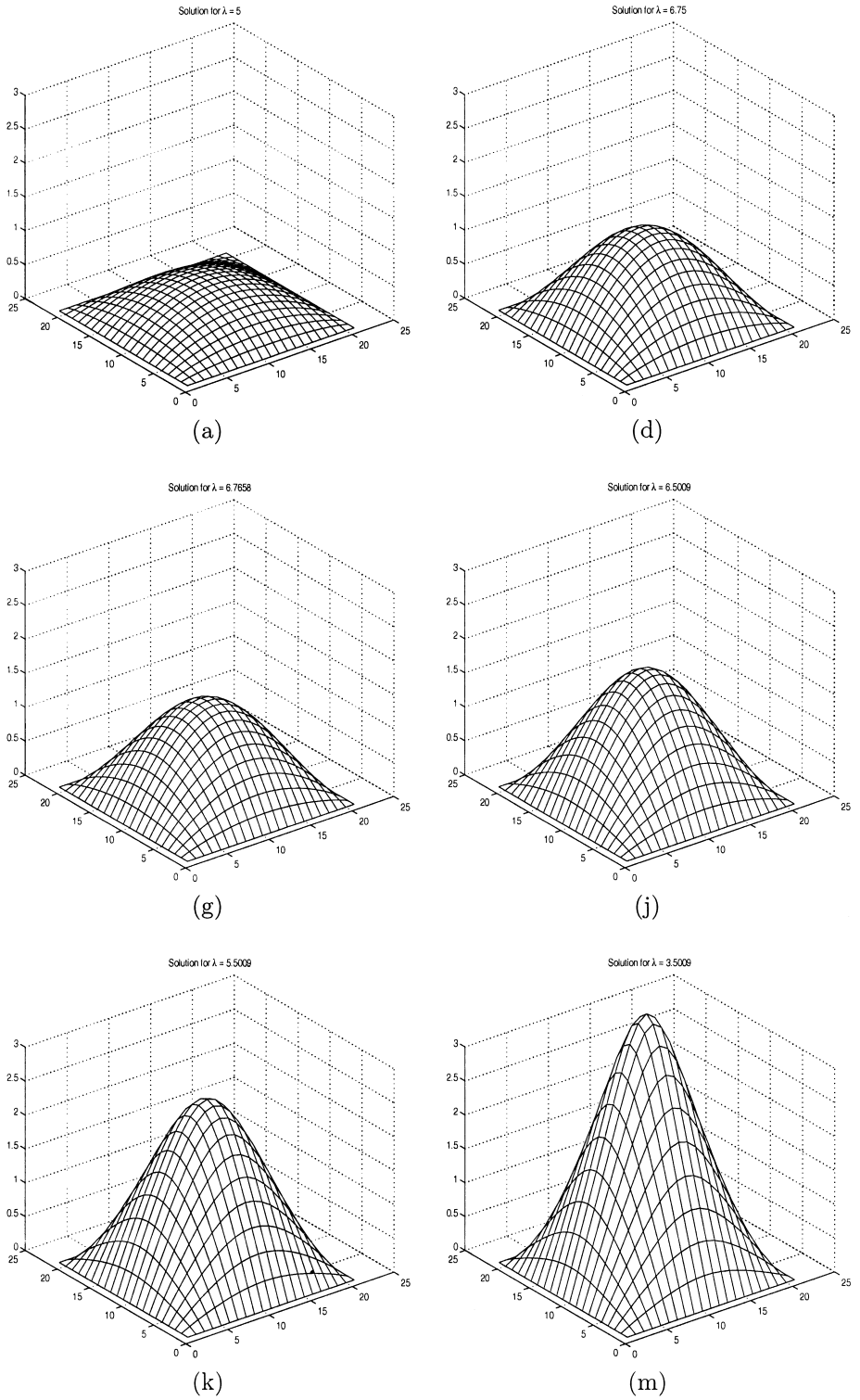


Fig. 3. Example 5.2: Solutions u associated with points on the graph of Fig. 2.

matrix–vector products required for the CG iterations are included in the counts of matrix accesses and matrix–vector products, respectively.

Fig. 2 shows the solution path computed. Solutions u associated with points marked on the solution path, corresponding to starred λ-values in Table 3, are plotted in Fig. 3.

The examples illustrate that the algorithms of the present paper can be used for path following in the presence of bifurcation and turning points. We remark that the number of matrix accesses required often decreases when the block-size r is increased, but the number of matrix–vector products typically increases with the block-size. Numerical examples that illustrate this for Algorithm 4.1 are reported in [5]. The selection of block-size should depend both on the problem at hand and on the architecture of the computer being used.

The step sizes $\Delta\lambda$, Δs and ε_b have been selected in a fairly arbitrary manner in the experiments. Implementation of a step-size control is likely to reduce the computational effort. We are presently investigating this issue.

6. Conclusion and future work

Experience from a large number of problems indicates that the algorithms presented in this paper are versatile tools for path following of large problems with symmetric Jacobians in an interactive computing environment. We are presently developing algorithms that are applicable to large-scale path following problems with nonsymmetric Jacobian matrices.

References

[1] E.L. Allgower, C.-S. Chien, K. Georg, C.-F. Wang, Conjugate gradient methods for continuation problems, J. Comput. Appl. Math. 38 (1991) 1–16.
[2] J. Baglama, D. Calvetti, L. Reichel, Iterative methods for computing a few eigenvalues of a large symmetric matrix, BIT 36 (1996) 400–421.
[3] J. Baglama, D. Calvetti, L. Reichel, Fast Leja points, Elec. Trans. Numer. Anal. 7 (1998) 124–140.
[4] J. Baglama, D. Calvetti, L. Reichel, A. Ruttan, Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling, J. Comput. Phys. 146 (1998) 203–226.
[5] D. Calvetti, L. Reichel, A block Lanczos method for large continuation problems, Numer. Algorithms 21 (1999) 109–118.
[6] D. Calvetti, L. Reichel, D.C. Sorensen, An implicitly restarted Lanczos method for large symmetric eigenvalue problems, Elec. Trans. Numer. Anal. 2 (1994) 1–21.
[7] D. Calvetti, L. Reichel, Q. Zhang, Conjugate gradient algorithms for symmetric inconsistent linear systems, in: J.D. Brown, M.T. Chu, D.C. Ellison, R.J. Plemmons (Eds.), Proceedings of the Cornelius Lanczos International Centenary Conference, SIAM, Philadelphia, 1994, pp. 267–272.
[8] C.-S. Chien, N.-H. Lu, Z.-L. Weng, Conjugate gradient methods for continuation problems II, J. Comput. Appl. Math. 62 (1995) 197–216.
[9] K.A. Cliffe, T.J. Garratt, A. Spence, Eigenvalues of the discretized Navier–Stokes equation with application to the detection of Hopf bifurcations, Adv. Comput. Math. 1 (1993) 337–356.
[10] D.W. Decker, H.B. Keller, Multiple limit point bifurcation, J. Math. Anal. Appl. 75 (1980) 417–430.
[11] T. Ericsson, A. Ruhe, The spectral transformation Lanczos method for the solution of large sparse symmetric generalized eigenvalue problems, Math. Comput. 35 (1980) 1251–1268.
[12] W.R. Ferng, C.T. Kelley, Mesh independence of matrix-free methods for path following, Report, Department of Mathematics, North Carolina State University, Raleigh, 1999.

[13] K. Georg, On tracing an implicitly defined curve by quasi-Newton steps and calculating bifurcation by local perturbation, SIAM J. Sci. Statist. Comput. 2 (1981) 35–50.
[14] J. Huitfeldt, Nonlinear eigenvalue problems – prediction of bifurcation points and branch switching, Report, Department of Computer Science, Chalmers University of Technology, Göteborg, 1991.
[15] J. Huitfeldt, A. Ruhe, A new algorithm for numerical path following applied to an example from hydrodynamics, SIAM J. Sci. Statist. Comput. 11 (1990) 1181–1192.
[16] H.B. Keller, Lectures on Numerical Methods in Bifurcation Problems, Springer, Berlin, 1987.
[17] D.C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
[18] Y.F. Zhou, A. Ruhe, Numerical path following and eigenvalue criteria for branch switching, in: J. Cullum, R.A. Willoughby (Eds.), Large Scale Eigenvalue Problems, Elsevier, Amsterdam, 1986, pp. 121–142.